BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 778 523 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
11.06.1997 Bulletin 1997/24

(51) Int Cl.[6]: **G06F 9/46**

(21) Application number: **96308797.8**

(22) Date of filing: **04.12.1996**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **06.12.1995 US 567822**

(71) Applicant: **XEROX CORPORATION**
Rochester New York 14644 (US)

(72) Inventors:
- **Webster, Marc W.**
  Rochester, NY 14607 (US)
- **Covert, David W.**
  Ontario, NY 14519 (US)
- **Rabjohns, Douglas T.**
  Fairport, NY 14450 (US)

(74) Representative: **Johnson, Reginald George et al**
Rank Xerox Ltd
Patent Department
Parkway
Marlow Buckinghamshire SL7 1YL (GB)

(54) Method of operation for an image processing apparatus

(57) The invention relates to a method of operation of an image processing apparatus having a controller (74) and a plurality of resources (104,106,108) arranged in an arbitrary configuration. Each of the resources (104,106,108) provides an associated processor storing data related to operational capabilities of the associated resource. The controller (74) is adapted to dynamically configure the image processing apparatus to operate in accordance with the operational capabilities of each of the processors by defining job requirements as a combination of images defining a set of sheets and specifying compilations of sheets. The job requirement is converted into an assembly tree relationship for merging into additional assembly trees for formulating the job requirement. The controller (74) responds to the data related to the operational capabilities of each of the modules (104,106,108) and to the assembly tree relationship of images, copy sheets, and compilations of copy sheets for providing a production tree relationship of the operational capabilities of the modules (104,106,108) including timing relationships for operating the image processing apparatus. The production tree relationship further permits arbitrary definition of a job requirement into a first segment independent of the capabilities of the modules and a second segment dependent upon selected capabilities of selected modules to allow the image processing apparatus to be partially configured based upon operator entered constraints. This technique further allows adaptive control of selective diagnostics.

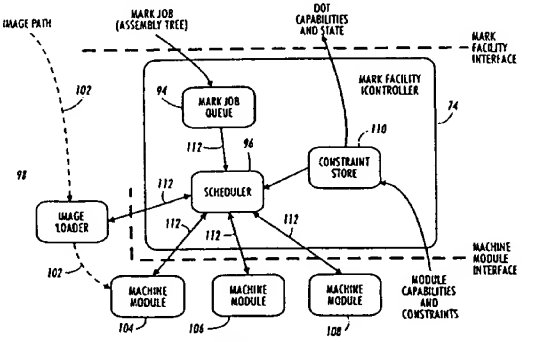

FIG. 7

EP 0 778 523 A2

to specify certain output terminal actions or selected diagnostic routines. Still another object of the present invention is to be able to define a document with an assembly tree component and a partial production tree component.

According to the present invention, an electronic image processing apparatus is provided with a controller and a plurality of resources in an arbitrary configuration. Each of the resources includes an associated processor storing data related to operational capabilities of the associated resource. The controller is adapted to dynamically configure the image processing apparatus to operate in accordance with the operational capabilities of each of the processors and to define processing requirements as a combination of images defining a set of sheets and specifying compilations of sheets. The processing requirement is converted into an assembly tree relationship for merging into additional assembly trees for formulating the job requirement. The controller responds to the data related to the operational capabilities of each of the modules and to the assembly tree relationship of images, copy sheets, and compilations of copy sheets for providing a production tree relationship of the operational capabilities of the modules including timing relationships for operating the image processing apparatus. The production tree relationship further permits arbitrary definition of a job requirement into a first segment independent of the capabilities of the modules and a second segment dependent upon selected capabilities of selected modules to allow the image processing apparatus to be partially configured based upon operator entered constraints. This technique further allows adaptive control of selective diagnostics.

The present invention will be described further, by way of examples, with reference to the accompanying drawings, in which:

Figure 1 is a simplified elevational view showing the relevant parts of a typical printing apparatus, on which the present invention may operate:

Figure 2 is a systems diagram showing a typical prior art machine configuration:

Figures 3, 4, and 5 illustrate arbitrary machine configurations capable of transparent control in accordance with the present invention:

Figure 6 illustrates a universal controller in accordance with the present invention:

Figure 7 illustrates a control architecture for the universal controller of Figure 6 in accordance with the present invention:

Figures 8 and 9 illustrate machine representations as transducers with constraints in accordance with the present invention:

Figure 10 illustrates a typical assembly tree configuration of a job requirement in accordance with the present invention:

Figure 11 illustrates the merging and transferring of assembly trees between facilities in accordance with the present invention: and

Figure 12 illustrates a typical production tree with constraints converted from an assembly tree to drive a printing apparatus in accordance with the present invention.

Figure 1 is a simplified elevational view of the paper path of an on-demand printing apparatus, capable of simplex or duplex output, in which a stream of digital video signals representative of images desired to be printed causes the desired images to be formed on a selected side of a print sheet. The particular architecture shown in Figure 1 is for an electrostatographic printer, but it will be understood that the principle of the invention could apply equally to other types of image-creation technologies, such as ink-jet printing. The printing apparatus, generally indicated as 10, contains one or more stacks of available sheets on which to print images, these stacks being indicated as 12a and 12b. The sheets of paper in the stacks 12a and 12b may differ in, for example, size, color, or the presence of a pre-printed letterhead. When it is desired to create an image on a sheet, a sheet of a desired type is drawn from a stack such as 12a or 12b, such as by respective feeders 14a, 14b, and the individual sheet is fed onto duplex loop 16.

Duplex loop 16 is typically in the form of an endless belt which is capable, by means of friction, static electricity, vacuum, or other means, of retaining a plurality of sheets thereon, thereby retaining a particular sheet until it is time for the sheet to receive an image on the side of the sheet facing outwardly from the belt of the duplex loop 16. In the architecture shown in Figure 1, it is intended that sheets "ride" on the outer surface of the belt of duplex loop 16. Along one portion of duplex loop 16, the belt of duplex loop 16 comes into close contact with a photoreceptor belt indicated as 18. At the point of close proximity of duplex loop 16 and photoreceptor belt 18, there may be provided a transfer corotron 20, the function of which will be familiar to skilled in the art of xerography.

In the xerographic-based embodiment of a printing apparatus shown in Figure 1, a device which shall be here generally referred to as an "imager" creates an electrostatic latent image on the surface of photoreceptor 18. Imager 22 has the function of receiving a sequence of digital signals representative of the desired image to be printed, and outputs a physical manifestation, such as a modulated laser scanning beam, to imagewise discharge selected areas on the photoreceptor 18 to create an electrostatic latent image representative of the image desired to be printed. As is known in the art of electrophotography, other stations along the path of photoreceptor 18, such as a charging bar and development unit (not shown) are also required to create the desired developed image on the photoreceptor belt 18. This developed image, which is typically in the form of a reverse image in toner particles on the photoreceptor

However, this is most likely still faster than developing an entirely new machine.

A Mark Facility Controller manages, coordinates, and represents the entire connected configuration of feeder modules, marker modules, finisher modules and output modules. These are referred to collectively as Machine Modules. There is one Mark Facility Controller for a configuration of machine modules. This collection of machine modules along with a Mark Facility Controller is referred to as a Document Output Terminal or alternatively, a Mark Facility.

The basic concept of the Mark Facility is that jobs coming from various sources such as decomposers, scanners, file systems, etc. can be sent to a common Mark Facility interface independent of where the job is coming from, and independent of what physical machine modules make up the DOT. The Mark Facility Controller is responsible for taking the mark job (described primarily via an Assembly Tree in accordance with the present invention) which is machine independent, mapping it on to the particular machine configuration present, and coordinating the machine modules to render the job.

Note that the Mark Facility Controller is not responsible for the Image Path. However, the Mark Facility Controller interacts closely with an Image Loader. The Image Loader is the demarcation point in the image path after which any further processing can be done in a deterministic amount of time. From the point of view of the Mark Facility Controller, the Image Loader acts as an "image flow control valve", and the Mark Facility Controller coordinates the "feeding of images" through the Image Loader along with and in the same manner as the feeding sheets.

Figure 6. illustrates how the Mark Facility Controller would interact with the various modules of the Mark Facility as well as the client(s) of the Mark Facility Interface. In particular, there is shown a Mark Facility controller 74 interconnected to arbitrary machine modules 76, 78, and 80 and image loader 82 by means of a page level control path shown as dashed lines at 84. Also shown connected to controller 74 are print server 86 and decompose facility 87 interconnected by means of either a service level control path, 83 or job level control path 88. Also, the image loader 82 can be connected between decompose facility 87 and a marker module, such as 78 by means of a page level image path, dotted lines, 90. Other operations such as a copy service, scan facility, and file system can also be part of the system. The diagram is an example configuration, not the required configuration. The service level control path 83 provides control of the entire Mark Facility (e.g. suspend the facility, resume the facility, submit mark job, cancel mark job, etc). The job level control path 88 would be used for streaming a job description (i.e. assembly tree): page level control 84 is essentially the scheduling of a page.

The Mark Facility Controller meets various requirements. In particular, the Mark Facility Controller ensures that the document output terminal produces what the operator asked for within the constraints of the DOT. If a jam or other anomaly (eg. crash) occurs during production then recovery must guarantee that no part of the output is lost or duplicated (e.g. can't lose or duplicate printed checks). The Mark Facility Controller ensures that the document output terminal is driven at rated speed whenever resources (paper, images) are available. This requirement implies that the Mark Facility Controller will control whatever buffering functions are necessary to ensure a steady supply of images to the marking module regardless of peculiarities of page order requirements of specific modules. It further implies that the Mark Facility Controller must be capable of streaming mark job (the job description coming into the Mark Facility Controller) to ensure uninterrupted delivery of prints.

In addition, the Mark Facility Controller must support a common Mark Facility Interface (software interface) for all DOTs: all DOTs are controlled through the same software interface, the Mark Facility Interface. The Mark Facility Controller must provide a uniform Machine Module Interface for marking, feeding and finishing for all devices supported by the architecture The Mark Facility Controller must provide information to enable Job Shop Scheduling (a.k.a. work flow management). This includes estimations of "time to complete job". This estimation includes factors like skipped pitches which can be predicted and perhaps those that can be statistically predicted: it does not account for unpredictable skipped pitches (e.g. unexpected jams). The Mark Facility Controller also provides information to its clients to enable load balancing of print jobs across multiple DOTs.

Also, the Mark Facility Controller makes available information about the DOT to Mark Facility clients, including information about the capabilities of the DOT and its current state. The Mark Facility Controller will not have any embedded knowledge about the client(s) of the Mark Facility Interface. That is there must not be any source dependencies incorporated into a Mark Facility Controller implementation. The Mark Facility Controller architecture has no a prior machine module specific knowledge. In particular, even if the Mark Facility Controller is physically packaged with a Marker Module, the Mark Facility Controller implementation software has no a priori knowledge of the marker module; it is completely independent of the marker module. Further, the underlying technology of the Mark Facility Controller should be machine module independent as well. Note that a particular implementation may be "tuned" or even "preset" for a certain set of configurations in order to decrease resource requirements.

The Mark Facility Controller has no knowledge of the image object content, processing requirements and representation. The Image Loader 82 is responsible for performing image processing and the hard real time buffering, synchronization and transmission of data between the Mark Facility Controller and the Marking Mod-

scheduler, at a much higher level) might query the Mark Facility Controller as to its ability to render a particular mark job. To validate the job without actually printing it, the bidding function would examine the machine graph (the description of the print machine configuration), validate that it could (or could not) do the job, and if so, submit a estimate of how long it would take, how soon it could start, etc. This would allow the print shop scheduler to perform load balancing or job pre-validation to improve shop productivity. Note that when estimating the time to produce a mark job, the Mark Facility Controller cannot take into account unforeseen circumstances like unexpected paper jams, etc.

While the Image Path is not part of the Mark Facility Controller, the Mark Facility Controller makes assumptions about its operation. The Mark Facility Controller treats images as work units to be scheduled (the same as sheets and compilations). The Mark Facility Controller expects modules sourcing images as work units to operate as proper Machine Modules and thus export the Machine Module Interface. Functions such as image buffer control, image processing, and consumption/generation of image formats are outside the scope of the Mark Facility Controller.

The Mark Facility Controller also needs to support offline finishing. There are two classifications of offline finishing: 1) completely independent standalone finishers and 2) configurations of feeders and/or finishers. A Mark Facility Controller is not needed in (1), but may be needed in (2). Therefore the Mark Facility Controller must be able to run on a platform suitable to this situation. For example, it may be desirable to have the Mark Facility Controller run on a laptop which can be connected to a configuration of feeders and finishers and coordinate them.

A system of machine modules is modeled as a collection of transducers which have constraints. as shown in Figures 8 and 9, that specify their behavior. Scheduling is accomplished by finding a sequence of transfers between the various transducer inputs and outputs that is consistent with the constraints. This is the essence that enables mix-and-match of Markers, Feeders, and Finishers. In particular, machine modules such as feeders. mark engines, and finishers are viewed as black boxes with portals which allow transfer units such as sheets, sets, plates, etc to enter or exit. Modules also have conceptual control signals which is used to specify desired capabilities such as simplex verses duplex, or staple verses bind. Modules export portals and control signals to the scheduler along with constraints. The constraints identify the subspace of signals that can be exhibited by the black-box on its portals. Every solution of the constraints corresponds to a feasible behavior of the black-box and every feasible behavior of the black-box corresponds to a solution of the constraints.

The scheduler creates a graph representing all of the modules.

For example, feeders 122, 124. and image loader 126 with specific constraints are connected to either or both of black and white mark engine 128 and color mark engine 130 as illustrated in Figure 8. Each of the mark engines includes specific constraints relative to a connection to complier/stapler 132 in turn connected to shrink wrapper 134 with associated constraints. Depending upon the interconnection and constraints, certain operations are acceptable and certain operations would fail. This interrelation can be illustrated by interconnected transducers illustrated in Figure 9. For example, transducer 136 is connected to transducer 148, transducer 136 responding to constraints 138, inputs 140 and control 142 to provide outputs 144 and 146. Output 146 is an input to transducer 148 in turn providing output 154 in response to control 152 and constraint 150.

When a print job is submitted, the scheduler creates a plan by solving the constraints and specifying the identity and times of transfer along the edges of a graph representation of module (the boundaries of the transducers). The machine module descriptions that the scheduler accepts are compositional, i.e. feeder, mark engine, and finisher descriptions can be merged at power-up time to form a single print machine description which can then be scheduled.

Capabilities are a means of describing what a DOT machine module can do such as feed paper, simple mark, staple. etc. Capabilities are described in terms of work units input. work units output, and the relationship between the inputs and the outputs using universally defined keywords. Traditional means of defining what a machine module can do have limited their description to the end outcome (stapled, bound, etc) which is insufficiently detailed to allow mix-and-match of markers, feeders, and finishers.

A capability is expressed on a transducer such as a machine, a machine module, or a component within a machine module. The capabilities defined what the transducer does. The capability identifies which kind of work unit is entering or exiting on which port of the transducer. It defines any constraints on timing of the work units (e.g. minimum 500 milliseconds between entering sheets), or attributes of the work units (e.g. paper size must be less than 17"), etc. It also defines the relationship between the inputs and outputs in terms of work unit properties e.g. finishing changed, added, deleted, etc., e.g. the sheet exiting has all the same properties as the sheet entering, except the orientation is changed from face up to face down).

The advantages over traditional methods of describing such data include the following. Traditionally, it was simply stated that a machine module with a stapler, merely stapled, and referred to the DPA ISO 10175 keyword STAPLE. However, actual machines differ widely in the physical details of accomplishing this operation. Therefore, by simply saying STAPLE does not provide enough information to determine whether a collection of machine modules can actually produce the requested

ments are to be produced (timing, production order, etc), only statements relating to the final output. The advantages over traditional methods of describing such data are that the assembly tree is highly expressive, machine independent, and extendible. The tree structure can express any kind of print engine job. This is a significant improvement over traditional specifications which often had to be altered whenever a product program needed to add a new job type. The assembly tree specification is independent of any particular machine, its configurations, and of it temporal constraints. This means the decomposer/print engine protocol doesn't have to change whenever the print engine changes. Because properties and finishing are expressed through keywords, new keywords can be added at any time, extending the space of expressible trees (traditionally, these have been hard-coded) The effect of the above is that the assembly tree enables true ESS/Print Engine plug-and-play.

In accordance with the present invention, an assembly tree is used for describing a universal canonical representation of a physical document. Because it is purely descriptive, an assembly tree can be used to describe a document that was scanned, a document that is to printed on a print engine, as well a document in intermediate steps between capture and mark. Traditionally different formats have been used at each stage because no format was expressive enough to cover all areas. Because the assembly tree simply describes a physical document rather than prescribing various actions to be taken, the assembly tree can be used in many interpretations. For example, when sent to a print engine, it can indicate a document to be printed. When received from a scanner, it can indicate a document that was scanned.

The genesis of an assembly tree is any input stream that a machine can accept. The input stream may be a PDL stream, an incoming Fax, or a stream of scanned images--each of these streams may be accompanied by some amount of 'job level' information. A Mark Facility and display are examples of consumers of assembly trees. Merge is an example of a component that consumes assembly trees and produces new assembly trees based on the input trees. The advantages over traditional methods of describing such data include having a common data format and a single canonical form representing documents. Having a single canonical form greatly simplifies system software, eliminating the needs for features such as conversations. Also, establishing a single canonical form for representing physical documents at the post-decomposed/image as bits creates new opportunities for "plug-in components". For example, a 3rd party vendor might make a "9-up" plug-in component that consumed assembly trees, and produced new assembly trees that were 9-up versions of the input assembly trees.

With reference to Figure 11 there is shown an example of the transfer of assembly tree representations between facilities. For example, block 230 illustrates the scanning of a document and conversion into the assembly tree format. At block 232, the scan document is transformed from a 1-up to a 4-up document by suitable manipulation of the assembly tree. The document or image in the assembly tree format can then be displayed as illustrated at block 234, sent for editing for incorporation into another electronic document as shown at 236 or merged with other assembly trees as illustrated 238. Block 238 is also shown as receiving decomposed images in assembly tree format at 242 and 240 to also be merged at block 238. The merged assembly tree can then be forwarded to a mark facility shown at 244 or filed as illustrated at 246.

In accordance with the present invention, there is provided a generic means, a production tree, to represent how a mark job is to be produced by a particular Document Output Terminal (DOT), a collection of markers, feeders, and finishers. The product tree structure is generic and can be used for any mark job on any DOT. Traditionally, such information was kept in ad hoc data structures customized to a particular DOT.

A production tree is a tree structure where the nodes represent capabilities of a DOT and the edges represent transfer of work units (images, sheets, and compilations) between various components of the DOT. The structure of the production tree establishes which capabilities of which machine module will be used to produce which part of the job: the timings on the edges establishes when capabilities are be to executed and in what order. A production tree is generally built by taking an input mark job in the form of an assembly tree discussed above, along with a model of the DOT expressed in terms of components with capabilities and mapping the assembly tree onto the model's capabilities.

Traditional methods of representing a document production plan generally have no means of accommodating a mix-and-match of various 3rd party markers, feeders, and finishers. The production tree enables this mix and match because of its generic means of relying on capabilities. The production tree is a central data structure in the Mark Facility architecture. Having a single representation across products allows for significant reuse among software that interacts with the production tree. in particular, a constraint-based scheduler.

Figure 12 illustrates a typical production tree to represent the manner of accomplishing a mark job requirement. Assume an assembly tree represents sheet 1 to receive image 1, sheet 2 to receive image 2, and sheet to receive image 3, the sheets 1, 2, and 3 to be compiled and stapled. A production tree representation of this generic assembly tree to achieve the result is illustrated. Specifically, Figure 12 illustrates the flow of the work units or images, sheets, and compilations are organized with appropriate timing indications to achieve the results. Block 250 illustrates sheet 1 feed at 3700 milliseconds and then image number 1 generated at 3800 milliseconds for marking at the appropriate marking ma-

3. A method of initiating a diagnostic procedure on an electronic image processing apparatus, the electronic image processing apparatus comprising a controller (74) and a plurality of modules (104,106,108), each of the modules including an associated processor, each of the processors storing data related to operational constraints of the associated module, a bus for interconnecting the processors to the controller for directing the operation of the modules, including:

receiving from each of the processors the data related to the operational constraints of each associated module,
defining each of the modules with inputs and outputs having said constraints,
interrogating each of the processors to determine the geometrical relationship of the interconnection of the modules, and
responding to the data related to the operational constraints of each of the processors and to the geometrical relationship of the interconnection of the modules for providing a timed sequence of matched inputs and outputs between modules to initiate the diagnostic procedure.

4. A method as claimed in claim 3, wherein the constraints include attribute constraints and/or time constraints.

5. A method as claimed in claim 3 or claim 4, wherein the step of initiating the diagnostic procedure is independent of a particular configuration of the plurality of modules, and/or wherein the step of providing a timed sequence of matched inputs and outputs between modules includes the step of providing electrical and mechanical events to accomplish the matched inputs and outputs.

6. A method of initiating a diagnostic routine in an electronic image processing apparatus comprising a controller and a plurality of machine modules, each of the modules including an associated processor electrically connected to the controller, each of the processors storing data related to operational capabilities of the associated module, including:

identifying a given diagnostic routine in an untimed format,
interrogating the machine modules to determine the interconnection and operational capabilities of each of the machine modules, including receiving data related to the operational constraints of each associated module,
analyzing the diagnostic routine in the untimed format and the operational constraints of each associated module,
converting the diagnostic routine in the untimed format into a timed relationship of a selected set of modules to initiate the diagnostic routine, and
coordinating the operation of the selected set of modules to complete the diagnostic routine.

7. A method as claimed in claim 6, including the step of defining each of the modules with inputs and outputs with said constraints and, optionally, providing a timed sequence of matched inputs and outputs between modules.

8. A method of operation of the image processing apparatus in a diagnostic mode, the image processing apparatus comprising a controller and a plurality of machine modules, each of the modules including an associated processor electrically connected to the controller, each of the processors storing data related to operational constraints of the associated module, including:

receiving in the controller a diagnostic requirement having a module independent element and a module specific element,
interrogating the machine modules to determine the interconnection of each of the machine modules,
converting the module independent and module specific elements into a selection of time compatible modules to support the diagnostic requirement and
coordinating the machine modules to complete the diagnostic requirement.

9. A method as claimed in claim 8, wherein the module independent and module specific elements are converted into a constraint and timing format for driving the machine modules.

10. A method of operation of the image processing apparatus to complete a given diagnostic request, the electronic image processing apparatus comprising a controller and a plurality of modules, each of the modules including an associated processor, each of the processors storing data related to operational constraints of the associated module, a bus for interconnecting the processors to the controller for directing the operation of the modules, including:

receiving from each of the processors the data related to the operational constraints of each associated module,
interrogating each of the processors to determine the geometrical relationship of the interconnection of the modules, and
responding to the diagnostic request and to the data related to the operational constraints of each of the processors for selecting a set of

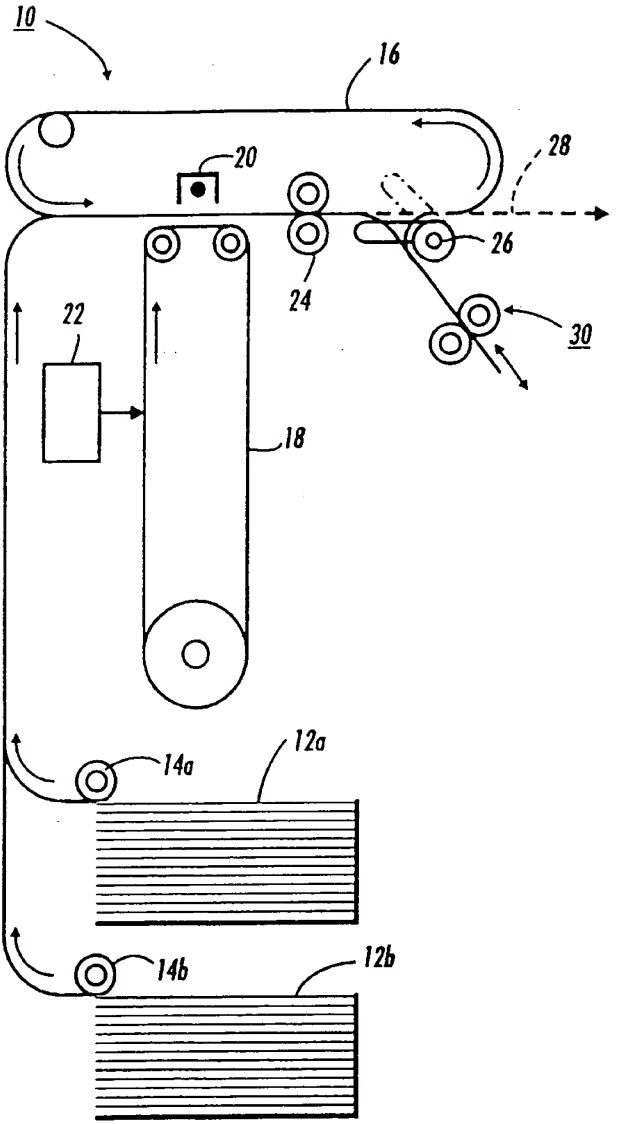

FIG. 1

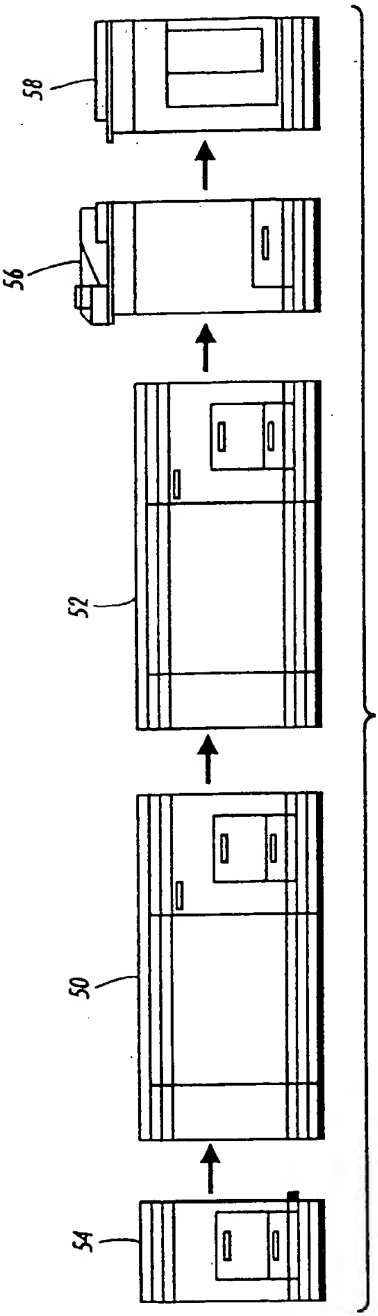

FIG. 3

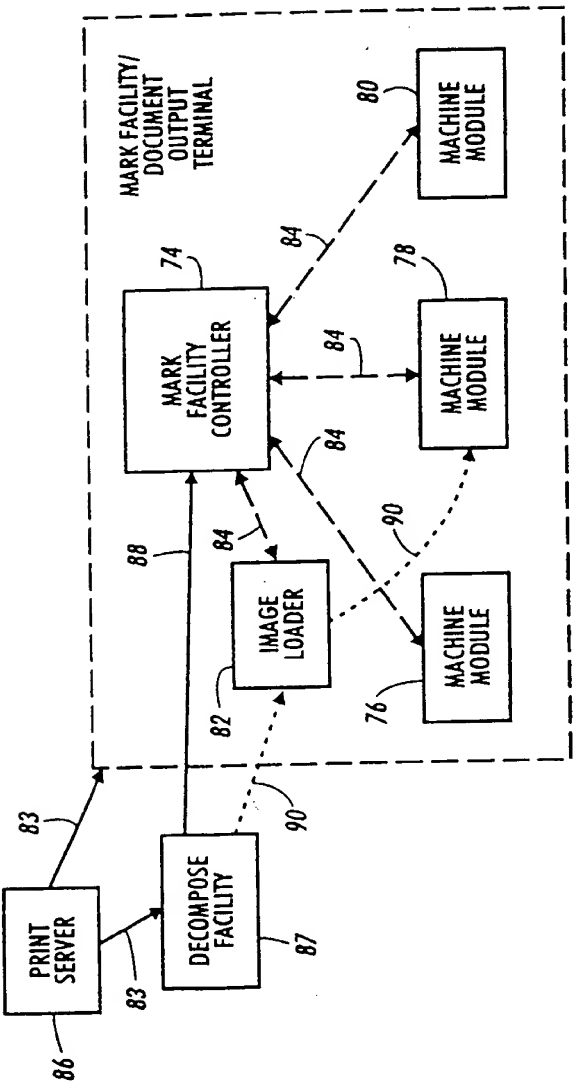
MARK FACILITY/
DOCUMENT
OUTPUT
TERMINAL
MARK
FACILITY
CONTROLLER
74
MACHINE
MODULE
80
MACHINE
MODULE
78
MACHINE
MODULE
76
IMAGE
LOADER
82
DECOMPOSE
FACILITY
87
PRINT
SERVER
86
83
84
88
90

FIG. 6

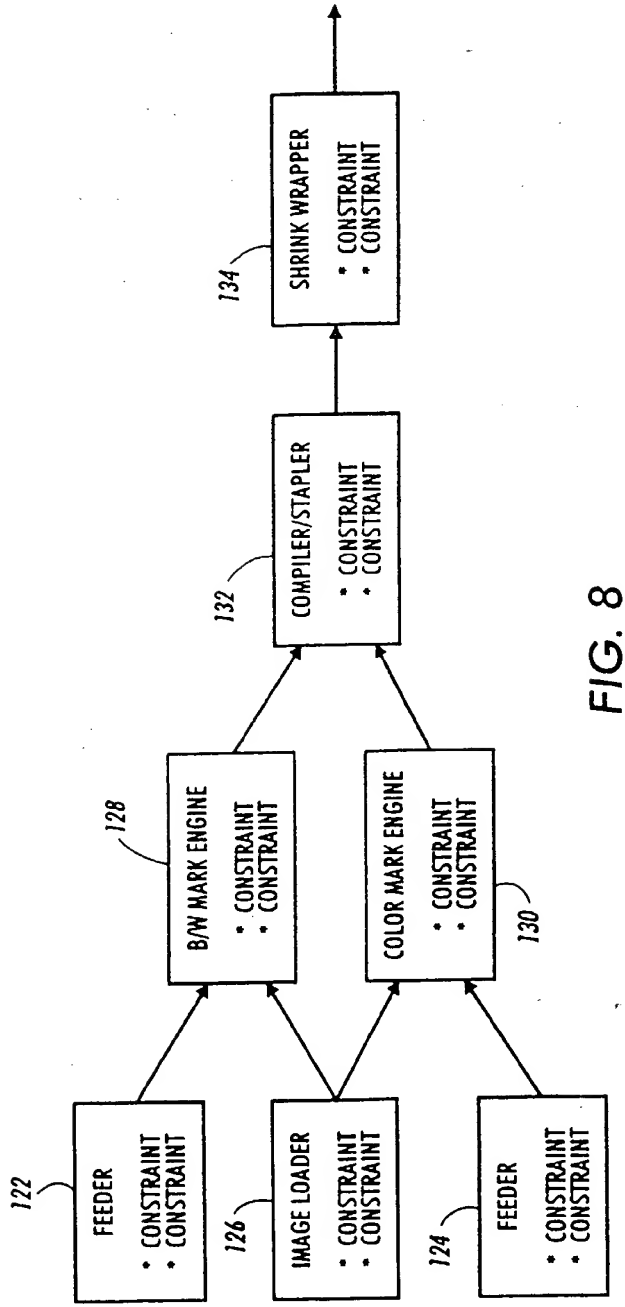
122
FEEDER
* CONSTRAINT
* CONSTRAINT
126
IMAGE LOADER
* CONSTRAINT
* CONSTRAINT
124
FEEDER
* CONSTRAINT
* CONSTRAINT
128
B/W MARK ENGINE
* CONSTRAINT
* CONSTRAINT
130
COLOR MARK ENGINE
* CONSTRAINT
* CONSTRAINT
132
COMPILER/STAPLER
* CONSTRAINT
* CONSTRAINT
134
SHRINK WRAPPER
* CONSTRAINT
* CONSTRAINT

FIG. 8

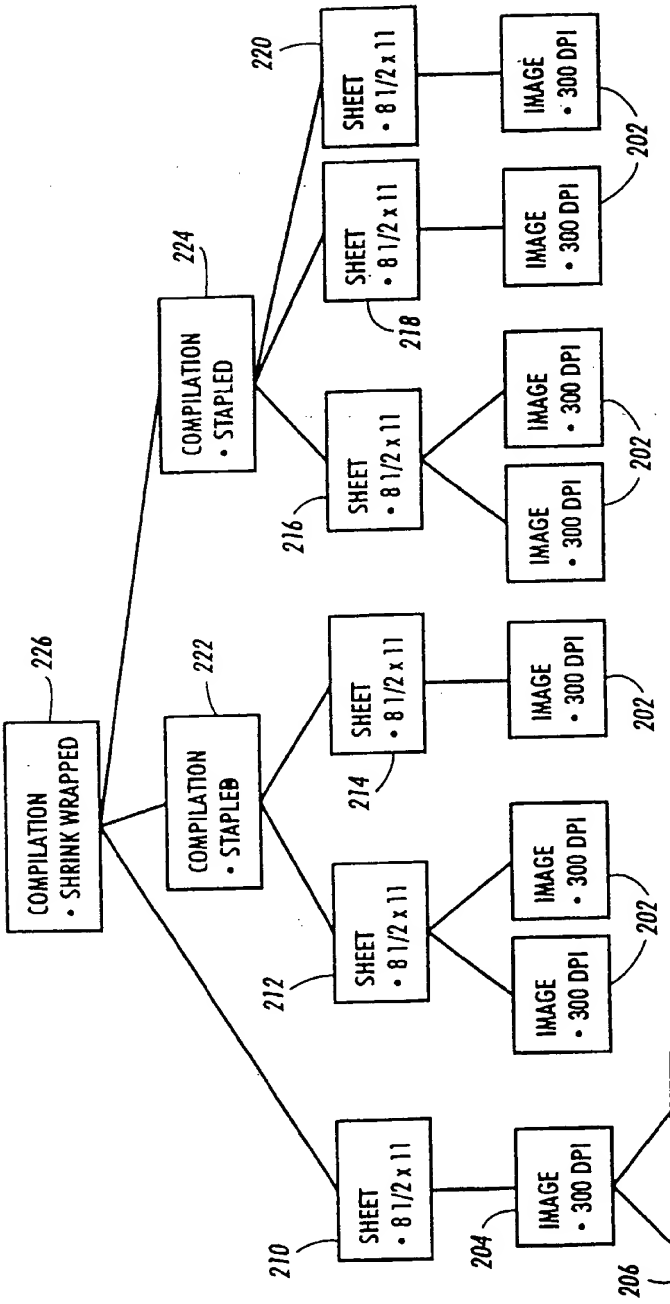
COMPILATION
• SHRINK WRAPPED
226
COMPILATION
• STAPLED
224
COMPILATION
• STAPLED
222
SHEET
• 8 1/2 x 11
220
SHEET
• 8 1/2 x 11
218
SHEET
• 8 1/2 x 11
216
SHEET
• 8 1/2 x 11
214
SHEET
• 8 1/2 x 11
212
SHEET
• 8 1/2 x 11
210
IMAGE
• 300 DPI
202
IMAGE
• 300 DPI
204
IMAGE
• 300 DPI
206
IMAGE
• 300 DPI
208

FIG. 10

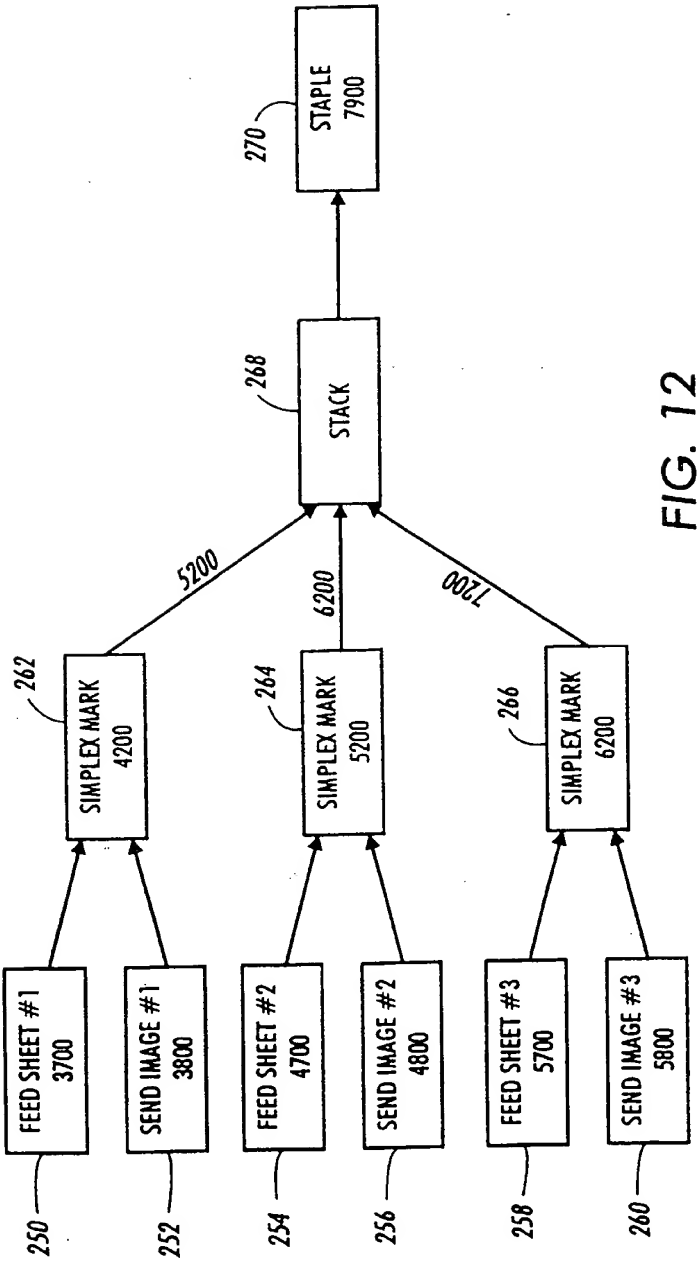
FEED SHEET #1
3700
SEND IMAGE #1
3800
FEED SHEET #2
4700
SEND IMAGE #2
4800
FEED SHEET #3
5700
SEND IMAGE #3
5800
SIMPLEX MARK
4200
SIMPLEX MARK
5200
SIMPLEX MARK
6200
STACK
STAPLE
7900
5200
6200
7200
250
252
254
256
258
260
262
264
266
268
270

FIG. 12